**IN THE CLAIMS:**

Please amend the claims as follows:

1. (Currently Amended) A system for dynamically linking application code created by a programmer into a running operating system kernel, comprising:

an environment library comprising one or more routines for insulating the application code from the operating system environment and for implementing a uniform execution environment, wherein the insulating of the application code allows the application code to perform a function by executing a routine that is standard to the application code instead of by using a routine that pertains to the running operating system kernel; and

a build system for constructing a loadable module from the application code and the environment library and for constructing a standard executable program from the loadable module and an execution library, wherein

the loadable module including a module input and a module output, and wherein

the execution library comprises one or more routines for transparently loading the loadable module into the running operating system kernel, passing arguments to the loadable module, and terminating and unloading the loadable module after receiving a termination signal, the one or more routines of the execution library setting up input/output channels by connecting a standard input and a standard output of the running operating system kernel to the module input and the module output.

2. (Original) The system of claim 1, further comprising an infrastructure library comprising one or more routines executed prior to loading the loadable module into the running operating system kernel and/or after unloading the loadable module from the kernel.

3. (Canceled)

4. (Previously presented) The system of claim 1, wherein the standard executable program may be in several files or a single file.

5. (Currently Amended) A method, comprising:

creating a loadable module, the loadable module including a module input and a module output;

creating an executable program; and

executing the executable program, wherein the executable program performs a method comprising the steps of:

setting up input/output channels by connecting a standard input and a standard output of a running operating system kernel to the module input and the module output;

inserting the loadable module into address space of the running operating system kernel, wherein, once the loadable the module is inserted into the address space, the loadable module begins to execute; and

waiting for the loadable module to connect via kernel/user channels and then connecting those kernel/user channels to the input/output channels, wherein the executable program is insulated from the running operating system kernel, and wherein the insulating of the executable program allows the executable program to perform a function by executing a routine that is standard to the executable program instead of by using a routine that pertains to the running operating system kernel.

6. (Previously Presented) The method of claim 5, wherein after the loadable module is inserted into the address space the loadable module performs a method comprising the steps of:

creating kernel/user channels;

creating a thread to execute application code; and

waiting for the thread to complete.

7. (Original) The method of claim 6, wherein the method performed by the loadable module further includes the step of freeing resources after the thread completes.

8-9. (Canceled)

10. (Previously presented) The system of claim 1, wherein one of the one or more routines of the execution library includes code for executing a utility for installing the loadable module into the running operating system kernel.

11. (Previously presented) The system of claim 10, wherein the utility for installing the loadable module into the running operating system kernel is the insmod program.

12. (Previously presented) The system of claim 1, wherein the build system includes instructions for compiling the application code into object code.

13. (Previously presented) The system of claim 12, wherein the build system further includes instructions for linking said object code with object code from the environment library to produce a linked object module.

14. (Previously presented) The system of claim 13, wherein the build system further includes instructions for converting the linked object module into a C code array.

15. (Previously presented) The system of claim 13, wherein the build system further includes instructions for compiling the C code array to produce an object file and for linking said object file with object code from the execution library to produce the standard executable program.

16. (Previously presented) The system of claim 1, wherein the environment library includes one or more routines to create kernel/user channels.

17. (Previously presented) The system of claim 1, wherein the environment library includes one or more routines to create a thread to execute the application code.

18. (Previously presented) The system of claim 17, wherein the environment library includes one or more routines for freeing resources and unloading the loadable module when the thread completes.

19. (Previously presented) The system of claim 1, wherein the environment library includes one or more routines for (a) copying in arguments; (b) creating communication channels that connect the loadable module to the executable program; (c) requesting a block of memory from the

operating system and storing a structure therein that describes the application code; and (d) putting data describing the application code on a task list.

20. (Previously presented) The system of claim 19, wherein the environment library further includes one or more routines for (a) removing said data describing the application code from the task list; (b) closing said communication channels that connect the loadable module to the executable program; and (c) freeing the block of memory that was requested from the operating system.

21. (Currently Amended) A computer readable storage medium including a set of instructions executable by a processor, the set of instructions operable to:

a first set of computer instructions for insulating application code from an operating system environment, wherein the insulating of the application code allows the application code to perform a function by executing a routine that is standard to the application code instead of by using a routine that pertains to the running operating system kernel;

a second set of computer instructions for constructing a loadable module from the application code and the first set of computer instructions, the loadable module including a module input and a module output; and

a third set of computer instructions for constructing an executable program from the loadable module and a fourth set of computer instructions; wherein

the fourth set of computer instructions includes computer instructions for transparently loading the loadable module into a running operating system kernel, passing arguments to the loadable module, and terminating and unloading the loadable module from the running operating system kernel after receiving a termination signal, the fourth set further includes at least one routine setting up input/output channels by connecting a standard input and a standard output of the running operating system kernel to the module input and the module output.

22. (Previously presented) The computer readable medium of claim 21, wherein the computer instructions for loading the loadable module into the running operating system kernel include computer instructions for executing a utility for installing the loadable module into the running operating system kernel.

23. (Previously presented) The computer readable medium of claim 22, wherein the utility for installing the loadable module into the running operating system kernel is the insmod program.

24. (Previously presented) The computer readable medium of claim 21, wherein the second set of computer instructions includes instructions for compiling the application code into object code.

25. (Previously presented) The computer readable medium of claim 24, wherein the second set of computer instructions further includes instructions for linking said object code with object code from the environment library to produce a linked object module.

26. (Previously presented) The computer readable medium of claim 25, wherein the third set of computer instructions includes instructions for converting the linked object module into a C code array.

27. (Previously presented) The computer readable medium of claim 26, wherein the third set computer instructions of further includes instructions for compiling the C code array to produce an object file and for linking said object file with object code from a library to produce the executable program.

28. (Previously presented) The computer readable medium of claim 21, wherein the first set of computer instructions includes instructions for creating kernel/user channels.

29. (Previously presented) The computer readable medium of claim 21, wherein the first set of computer instructions includes instructions for creating a thread to execute the application code.

30. (Previously presented) The computer readable medium of claim 29, wherein the first set of computer instructions includes instructions for freeing resources and unloading the loadable module when the thread completes.

31. (Previously presented) The computer readable medium of claim 21, wherein the first set of computer instructions includes instructions for (a) creating communication channels that connect the loadable module to the executable program; (b) requesting a block of memory from the operating system and storing a structure therein that describes the application code; and (c) putting data describing the application code on a task list.

32. (Previously presented) The computer readable medium of claim 31, wherein the first set of computer instructions further includes instructions for (a) removing said data describing the application code from the task list; (b) closing said communication channels that connect the loadable module to the executable program; and (c) freeing the block of memory that was requested from the operating system.

33. (Previously presented) The computer readable medium of claim 21, wherein the executable program may be in several files or a single file.

34. (Currently Amended) A computer system, comprising:

first means for insulating application code from an operating system environment, wherein the insulating of the application code allows the application code to perform a function by executing a routine that is standard to the application code instead of by using a routine that pertains to the running operating system kernel;

second means for constructing a loadable module from the application code and the first means, the loadable module including a module input and a module output;

third means for constructing an executable program from the loadable module; and

fourth means for transparently loading the loadable module into a running operating system kernel, passing arguments to the loadable module, and terminating and unloading the loadable module from the running operating system kernel after receiving a termination signal, the fourth means includes at least one routine setting up input/output channels by connecting a standard input and a standard output of the running operating system kernel to the module input and the module output.

35. (Previously presented) The computer system of claim 34, wherein means for loading the loadable module into the running operating system kernel include means for executing a utility for installing the loadable module into the running operating system kernel.

36. (Previously presented) The computer system of claim 35, wherein the utility for installing the loadable module into the running operating system kernel is the insmod program.

37. (Previously presented) The computer system of claim 34, wherein the second means includes means for compiling the application code into object code.

38. (Previously presented) The computer system of claim 37, wherein the second means further includes means for linking said object code with object code from the environment library to produce a linked object module.

39. (Previously presented) The computer system of claim 38, wherein the third means includes means for converting the linked object module into a C code array.

40. (Previously presented) The computer system of claim 39, wherein the third means further includes instructions for compiling the C code array to produce an object file and for linking said object file with object code from a library to produce the executable program.

41. (Previously presented) The computer system of claim 34, wherein the first means includes means for creating kernel/user channels.

42. (Previously presented) The computer system of claim 34, wherein the first means includes means for creating a thread to execute the application code.

43. (Previously presented) The computer system of claim 42, wherein the first means includes means for freeing resources and unloading the loadable module when the thread completes.

44. (Previously presented) The computer system of claim 34, wherein the first means includes means for (a) creating communication channels that connect the loadable module to the executable program; (b) requesting a block of memory from the operating system and storing a structure therein that describes the application code; and (c) putting data describing the application code on a task list.

45. (Previously presented) The computer system of claim 44, wherein the first means further includes means for (a) removing said data describing the application code from the task list; (b) closing said communication channels that connect the loadable module to the executable program; and (c) freeing the block of memory that was requested from the operating system.

46. (Previously presented) The computer system of claim 34, wherein the executable program may be in several files or a single file.

47. (Currently Amended) A computer system for dynamically linking application code created by a programmer into a running operating system kernel, comprising:

means for creating a loadable module, the loadable module including a module input and a module output; and

means for creating an executable program that is configured to performs a method comprising the steps of:

setting up input/output channels by connecting a standard input and a standard output of the running operating system kernel to the module input and the module output;

inserting the loadable module into address space of the running operating system kernel, wherein, once the loadable the module is inserted into the address space, the loadable module begins to execute; and

waiting for the loadable module to connect via kernel/user channels and then connecting those kernel/user channels to the input/output channels, wherein the executable program is insulated from the running operating system kernel, and wherein the insulating of the executable program allows the executable program to perform a function by executing a routine that is standard to the executable program instead of by using a routine that pertains to the running operating system kernel.

48. (Previously presented) The computer system of claim 47, wherein the loadable module is configured to perform a method after the loadable module is inserted into the operating system address space, wherein said method comprises the steps of:

creating kernel/user channels;

creating a thread to execute the application code; and

waiting for the thread to complete.

49. (Previously presented) The computer system of claim 48, wherein the method performed by the loadable module further includes the step of freeing resources after the thread completes.

50. (Previously presented) The computer system of claim 47, wherein the step of inserting the loadable module into an operating system address space includes the step of creating a child process, wherein the child process replaces its image with the insmod process image.

51. (Previously presented) The computer system of claim 50, wherein the step of inserting the loadable module into an operating system address space further includes the step of piping the loadable module to the insmod process.

52. (Currently Amended) A method for dynamically linking application code created by a user into a running operating system kernel, comprising:

constructing a loadable module from application source code written by a user, the loadable module including a module input and a module output;

setting up input/output channels by connecting a standard input and a standard output of the running operating system kernel to the module input and the module output;

creating an executable program, wherein the executable program is configured to transparently load the loadable module into the running operating system kernel;

executing the executable program, thereby loading the loadable module into the running operating system kernel; and

unloading the loadable module from the running operating system kernel by sending a termination signal to the executable program, wherein the executable program is insulated from the running operating system kernel, and wherein the insulating of the executable program allows the executable program to perform a function by executing a routine that is standard to the executable program instead of by using a routine that pertains to the running operating system kernel.

53. (Previously presented) The method of claim 52, wherein the application source code is an ordinary application program.

54. (Previously presented) The method of claim 52, wherein the step of constructing the loadable module from the application source code consists essentially of executing a pre-defined makefile.

55. (Previously presented) The method of claim 52, further comprising the step of providing a makefile to the user, wherein the user performs the step of constructing the loadable module by executing the makefile after the user has created the application code.

56. (Previously presented) The method of claim 52, further comprising the step of providing the user with a library comprising object code, wherein the step of constructing the loadable module from the application source code comprises the steps of compiling the application source code into object code; linking the object code with object code from the library to produce a linked object module; and converting the linked object module into a C code array.

57. (Previously presented) The method of claim 56, wherein the step of constructing the loadable module further comprises the step of compiling the C code array to produce an object file.

58. (Previously presented) The method of claim 57, further comprising the step of providing the user with a second library comprising object code, wherein the step of constructing the executable program comprises the steps of linking the object file with object code from the second library.

59. (Previously presented) The method of claim 56, wherein the library includes one or more routines to create kernel/user channels.

60. (Previously presented) The method of claim 56, wherein the library includes one or more routines to create a thread to execute the application code.

61. (Previously presented) The method of claim 60, wherein the library includes one or more routines for freeing resources and unloading the loadable module when the thread completes.

62. (Previously presented) The method of claim 56, wherein the library includes one or more routines for (a) copying in arguments; (b) creating communication channels that connect the loadable module to the executable program; (c) requesting a block of memory from the operating system and storing a structure therein that describes the application code; and (d) putting data describing the application code on a task list.

63. (Previously presented) The method of claim 62, wherein the environment library further includes one or more routines for (a) removing said data describing the application code from the task list; (b) closing said communication channels that connect the loadable module to the executable program; and (c) freeing the block of memory that was requested from the operating system.

64. (Previously presented) The method of claim 52, wherein the executable program is configured to set up input/output channels.

65. (Previously presented) The method of claim 52, wherein the executable program is configured to execute a utility for installing the loadable module into the running operating system kernel.

66. (Previously presented) The method of claim 65, wherein the utility for installing the loadable module into the running operating system kernel is the insmod program.

67. (Previously presented) The method of claim 65, wherein the step of inserting the loadable module into an operating system address space includes the step of creating a child process, wherein the child process replaces its image with the insmod process image.

68. (Previously presented) The method of claim 67, wherein the step of inserting the loadable module into an operating system address space further includes the step of piping the loadable module to the insmod process.